Kubernetes Scaling SIG (K8Scale)

Aaron Crickenberger (on behalf of Bob Wise)

Samsung SDS Research America

SAMSUNG SDS SAMSUNG

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

This presentation is intended to provide information concerning Samsung's efforts around containers and container orchestration. We do our best to make sure that information presented is accurate and fully up-to-date. However, the presentation may be subject to technical inaccuracies, information that is not up-to-date or typographical errors. As a consequence, Samsung does not in any way guarantee the accuracy or completeness of information provided on this presentation. Samsung reserves the right to make improvements, corrections and/or changes to this presentation at any time.

The information in this presentation or accompanying oral statements may include forward-looking statements. These forward-looking statements include all matters that are not historical facts, statements regarding the Samsung Data System' intentions, beliefs or current expectations concerning, among other things, market prospects, growth, strategies, and the industry in which Samsung operates. By their nature, forward-looking statements involve risks and uncertainties, because they relate to events and depend on circumstances that may or may not occur in the future. Samsung cautions you that forward looking statements are not guarantees of future performance and that the actual developments of Samsung, the market, or industry in which Samsung operates may differ materially from those made or suggested by the forward-looking statements contained in this presentation or in the accompanying oral statements. In addition, even if the information contained herein or the oral statements are shown to be accurate, those developments may not be indicative developments in future periods.

Logos remain the property of their respective owners. So there.

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Kubernetes 101

Goals and Ideals

Some Stories

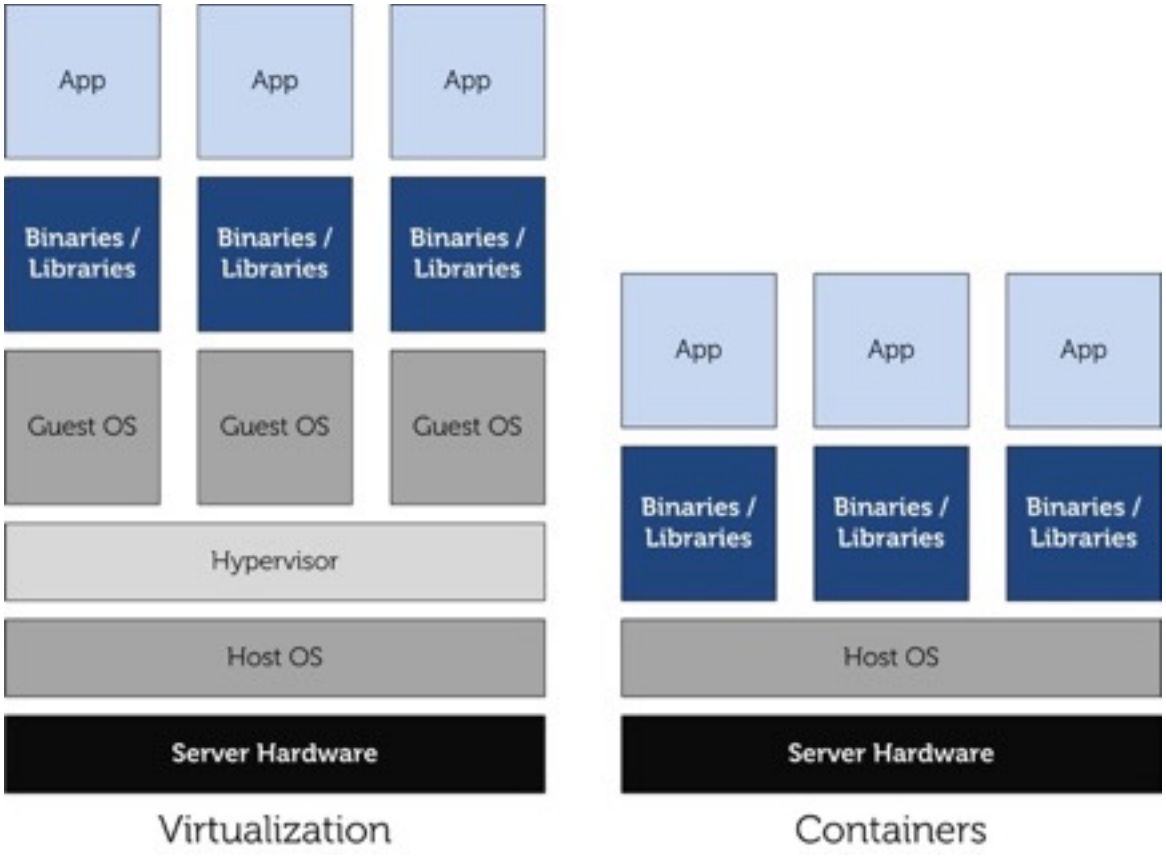
Looking Ahead

Challenges

Why K8scale?

SAMSUNG SDS SAMSUNG

The Same Slide You Always See In Any Talk About Containers



Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Containers Are Great, But, You Know…

- CAGBY, they die all the time
- CAGBY, wiring them up is boilerplate
- CAGBY, how do they network with the rest of the world
- CAGBY, how can they implement stateful services
- CAGBY, there are way more of them than VM's
- CAGBY, there's an awful lot of Kool-aid to drink

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Kubernetes

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

The Same Question Everyone Always Asks About Kubernetes

- It’s Greek for “helmsman”
- It’s also the root of the words “governor” and “cybernetic”

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Cybernetic Governor Helmsman 101

Goals and Ideals

Some Stories

Looking Ahead

Challenges

Why K8scale?

SAMSUNG SDS SAMSUNG

Kubernetes Core Concepts

- Labels / Annotations / Namespaces
- Nodes / Pods
- Services
- Replication Controller / Replica Sets

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

Kubernetes Core Concepts

- Labels
  - key/value pairs (eg: tier=frontend)
- Annotations
  - allow for structured metadata
- Namespaces
  - logically separate resources

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Kubernetes Core Concepts

- Nodes
  - hosts capable of running containers
- Pods
  - co-located groups of containers
  - share same networking space
  - scheduled together as a unit
  - each Pod gets its own IP

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Kubernetes Core Concepts

- Services
  - a logical set of pods that implement some service
  - set is chosen via label selector
  - a policy on how those pods may be accessed
  - decouple service lifecycle from pod lifecycle

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

Kubernetes Core Concepts

- Replication Controller
  - ensure N pod ‘replicas’ are running
  - healing: reschedule if pod dies
  - scaling: change N from 1 to 100
- Replica Set
  - same as above, but ‘replicas’ are now defined by a label selector

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

Kubernetes Extra Concepts

- ~~Labels / Annotations / Namespaces~~
- ~~Nodes / Pods~~
- ~~Services~~
- ~~Replication Controller / Replica Sets~~
- Secrets
- Deployments
- Horizontal Pod Autoscaling
- Jobs
- Daemon Sets
- Volumes / Persistent Volumes

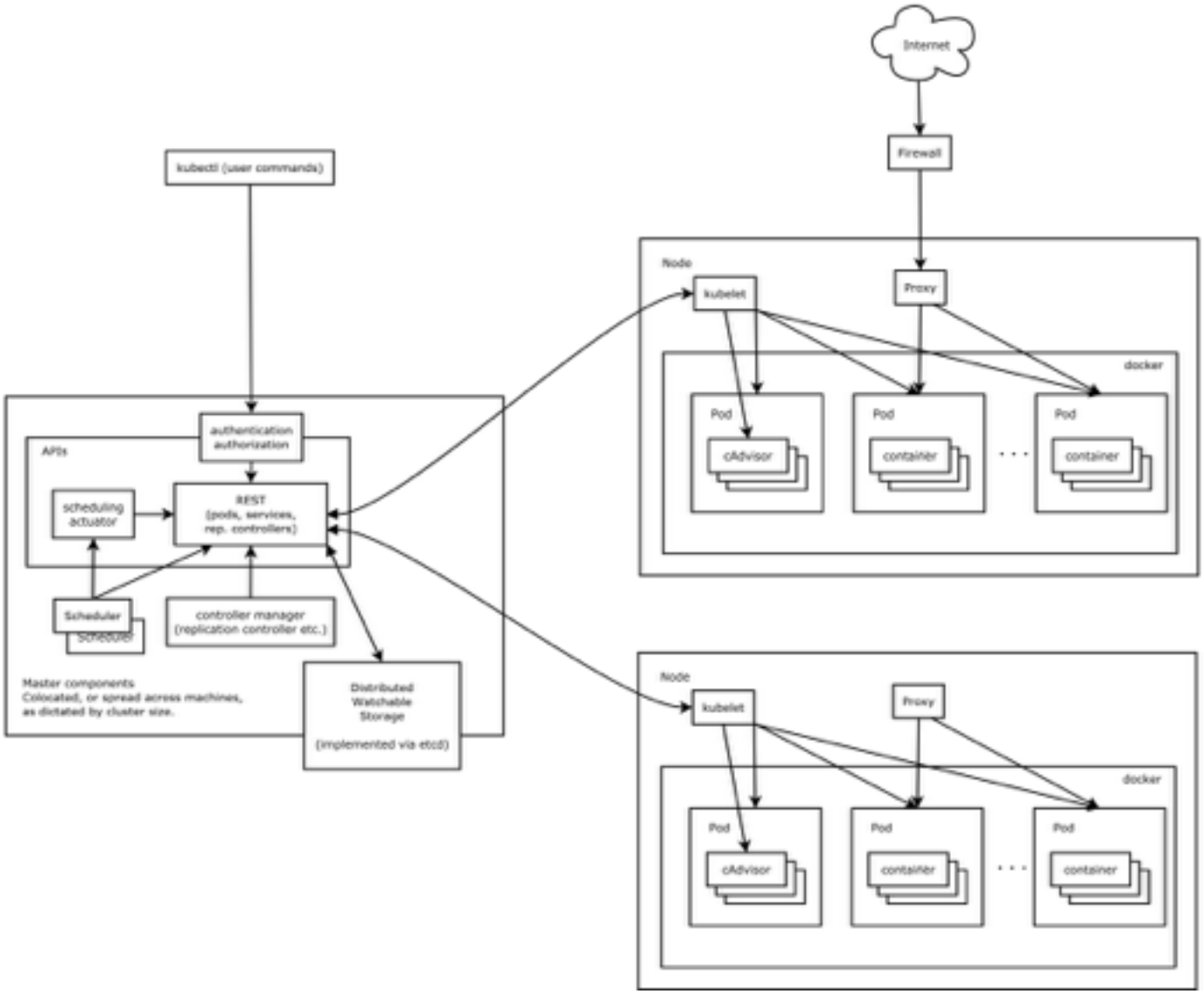
Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Kubernetes Core Components

- Master
  - kube-apiserver
  - kube-controller-manager
  - kube-scheduler
- Node
  - kubelet
  - kube-proxy
- Registry
  - etcd

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Kubernetes Architecture



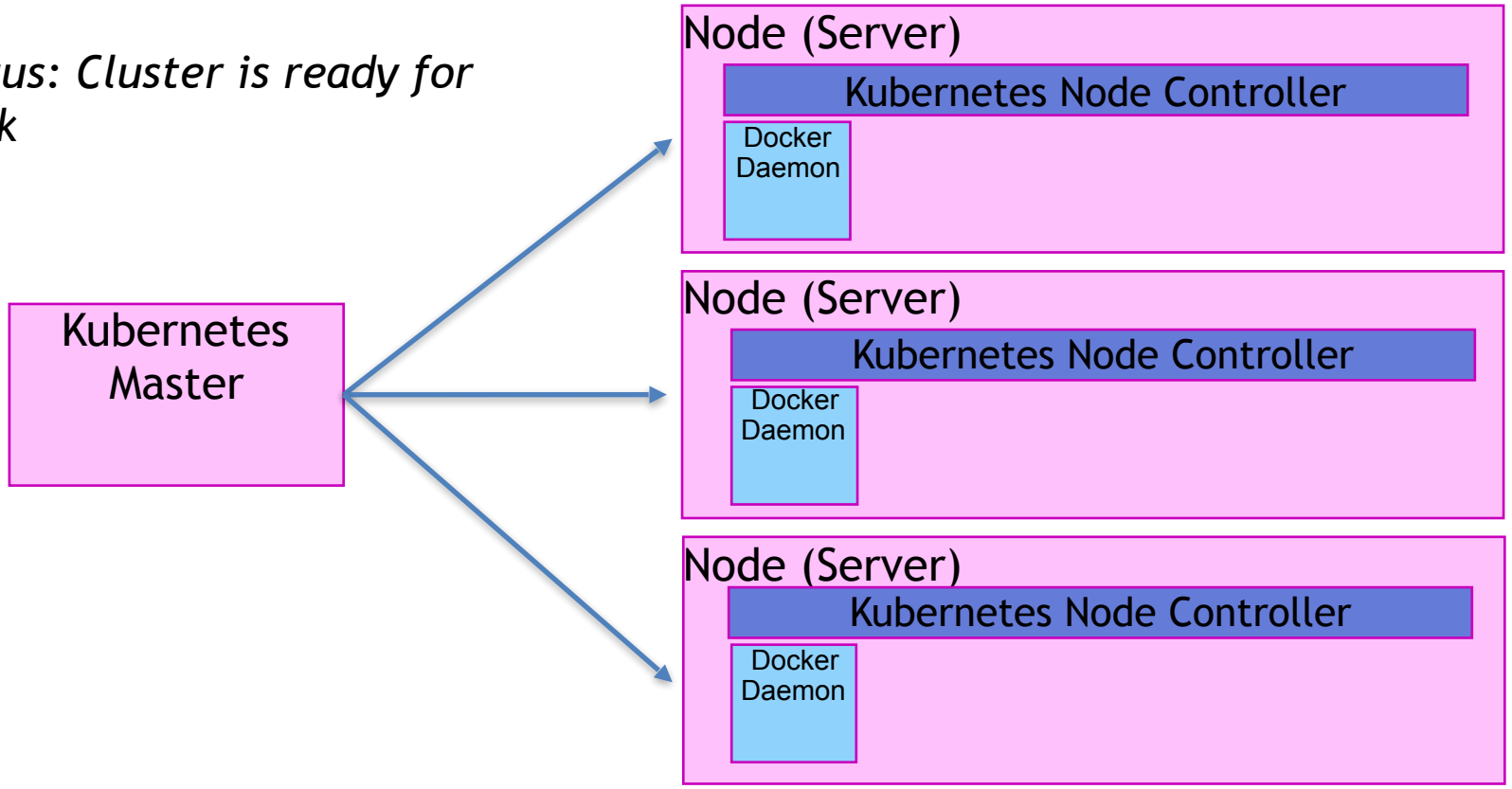
Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS

SAMSUNG

Example

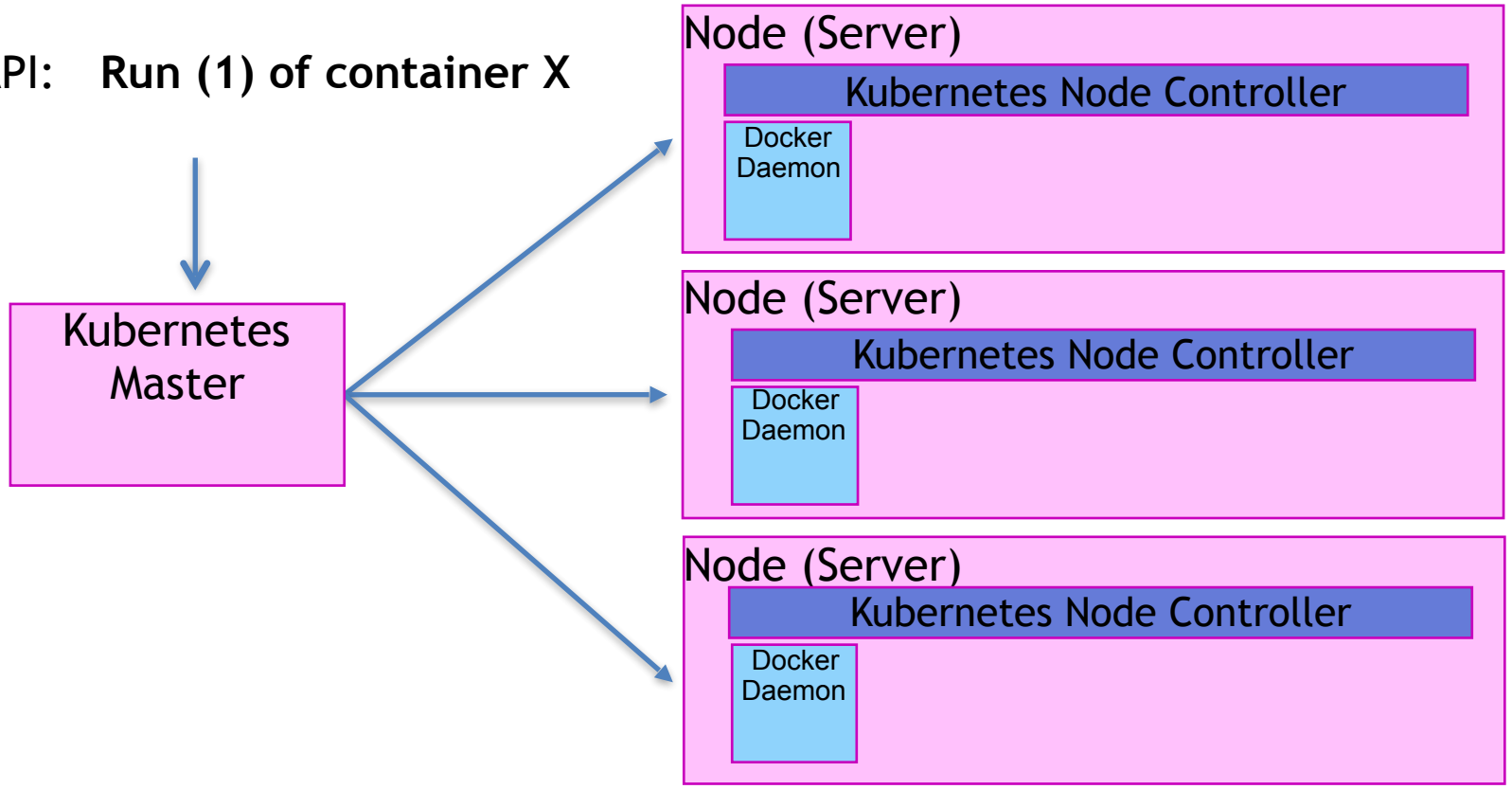
Status: Cluster is ready for work



Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Example

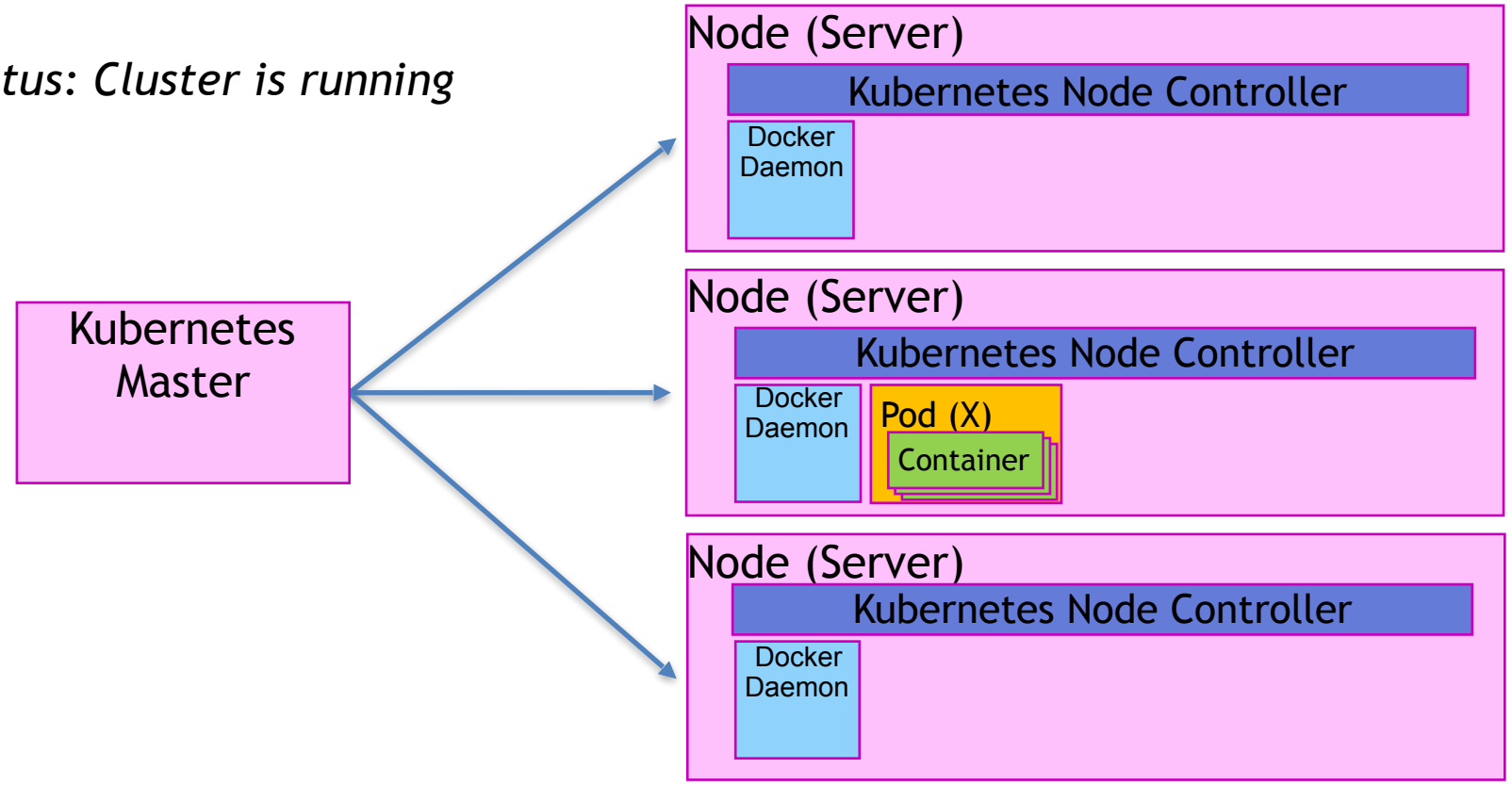
To API: **Run (1) of container X**



Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Example

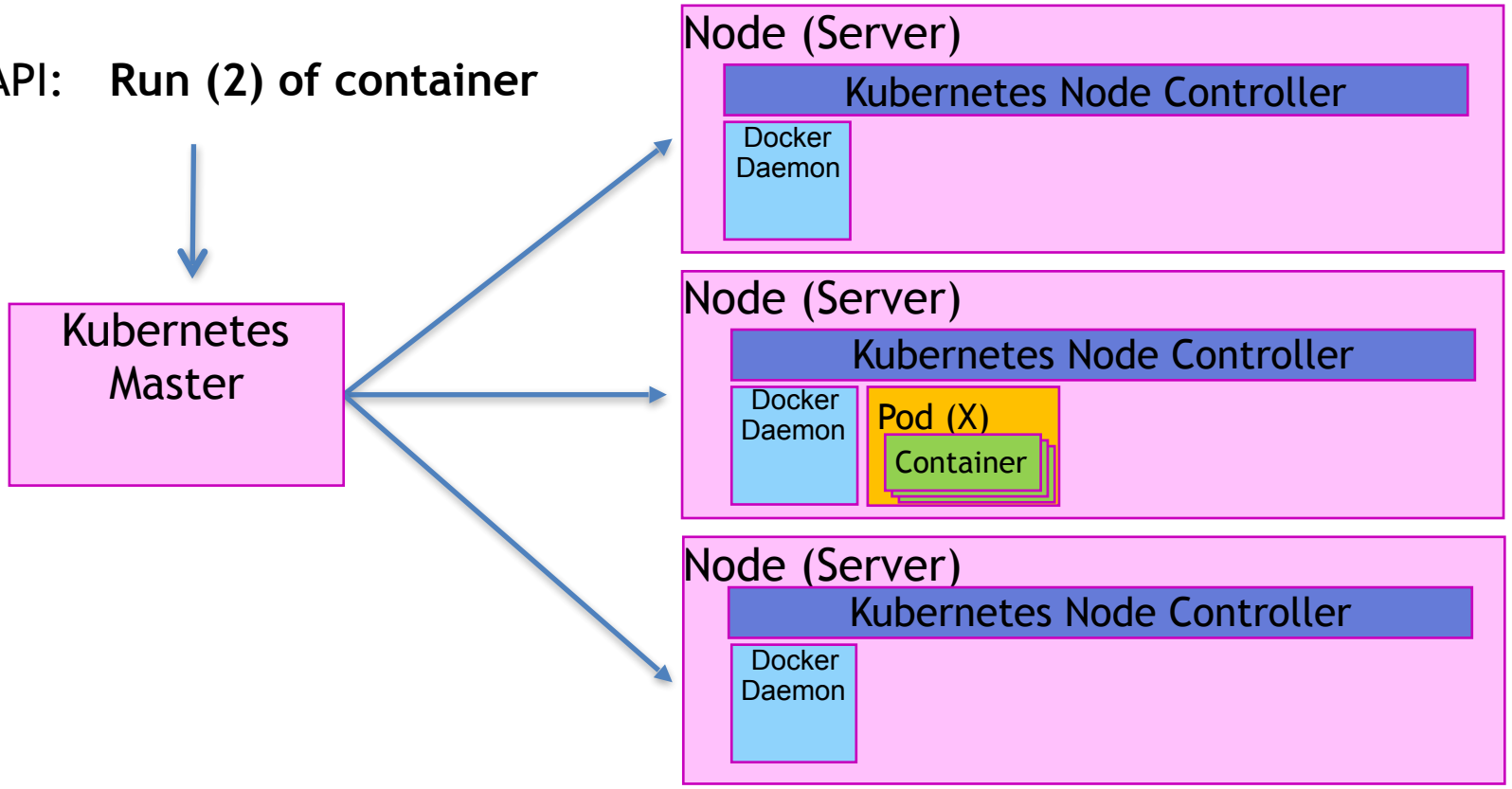
Status: Cluster is running
X



Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Example

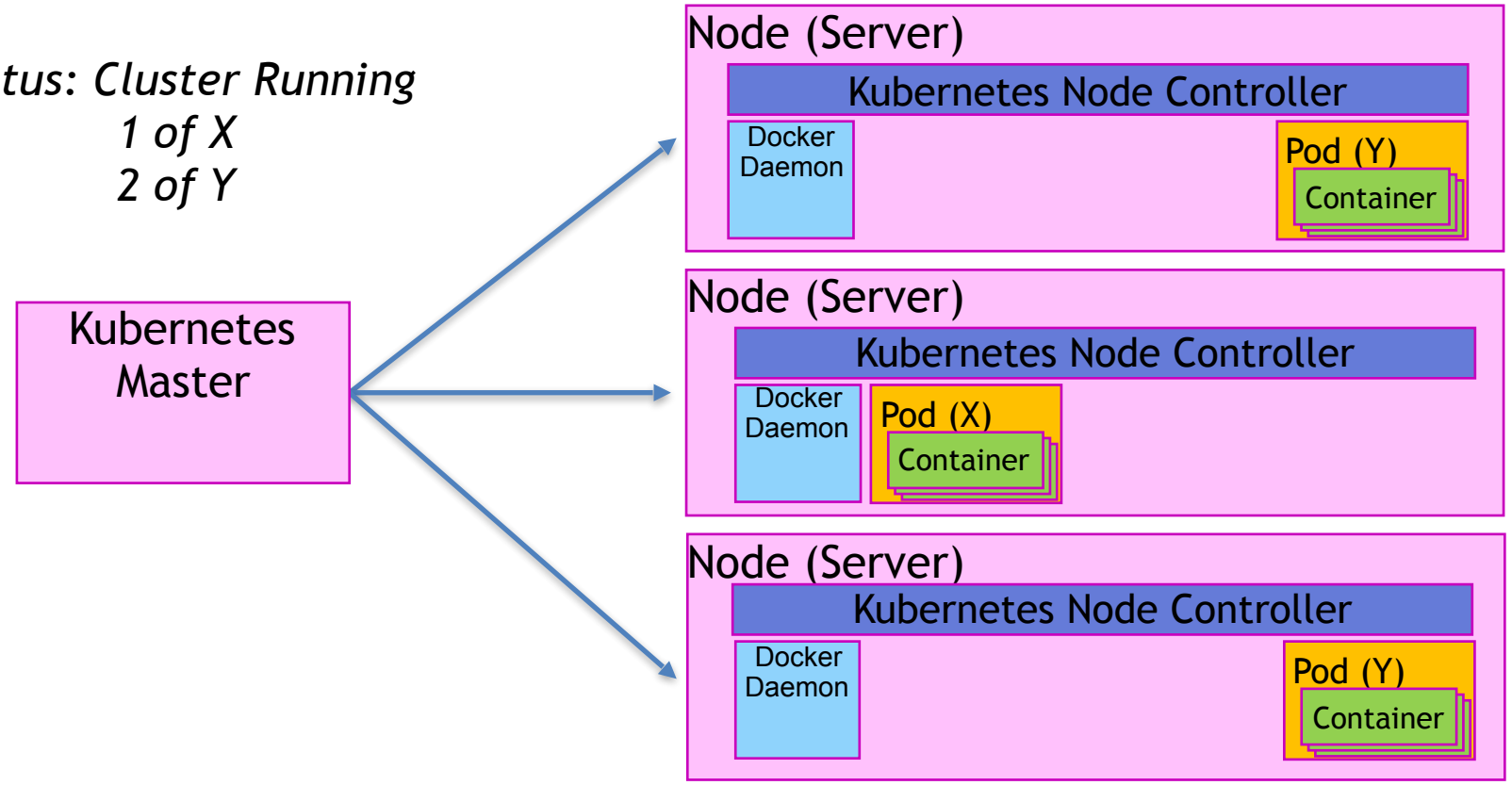
To API: **Run (2) of container Y**



Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Example

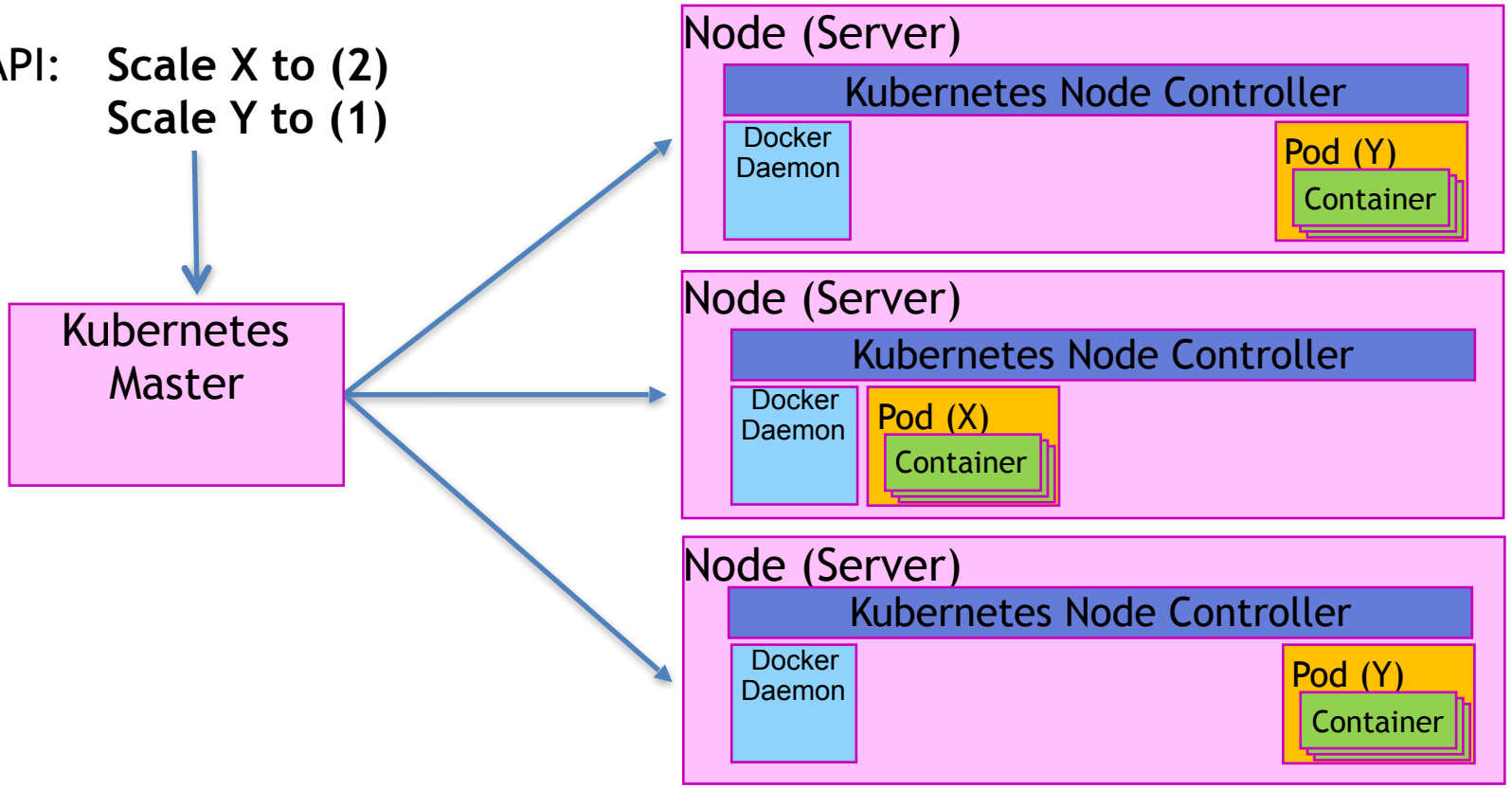
Status: Cluster Running
1 of X
2 of Y



Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Example

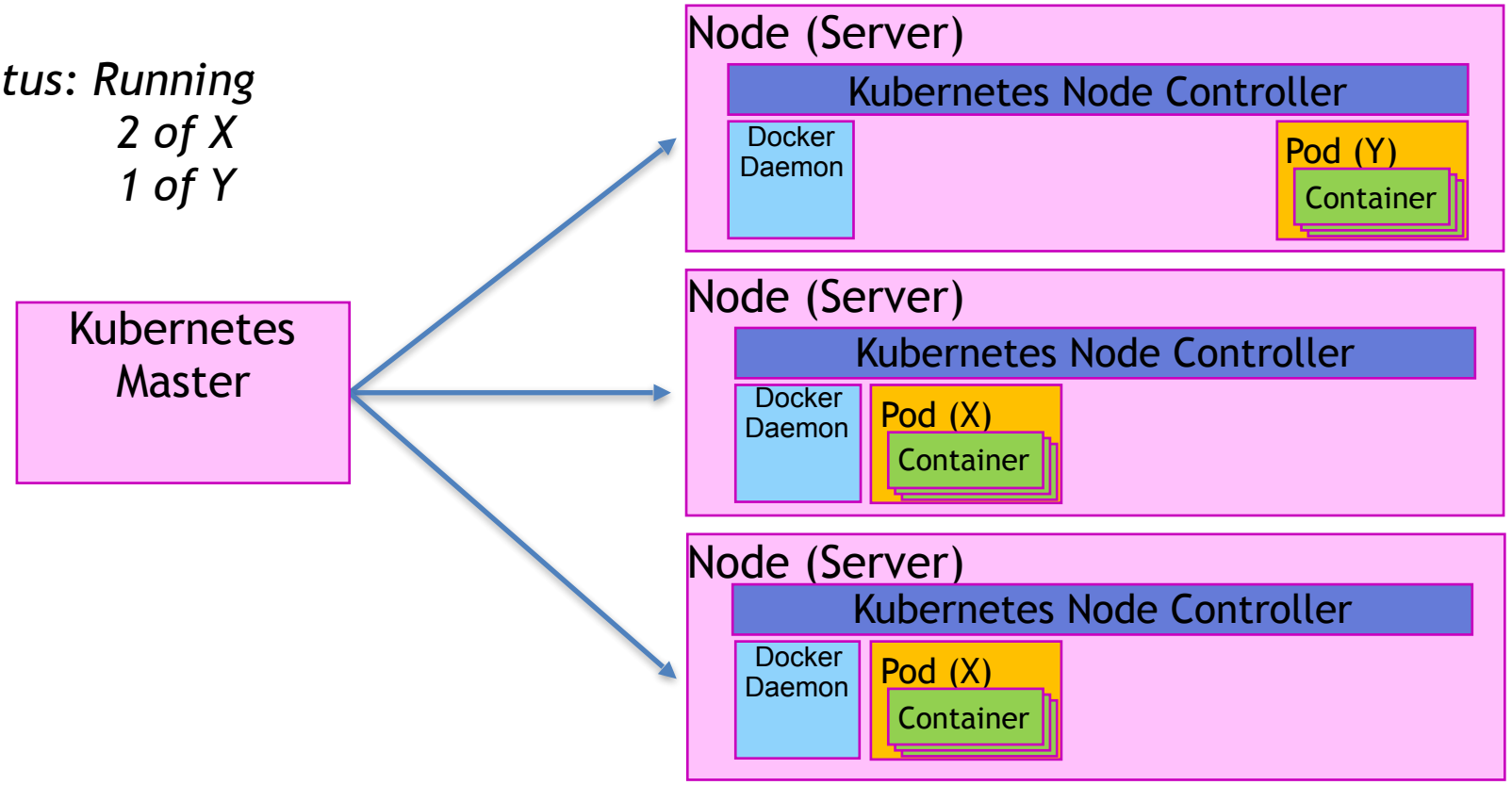
To API: **Scale X to (2)**
Scale Y to (1)



Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved
SAMSUNG SDS SAMSUNG

Example

Status: Running
2 of X
1 of Y



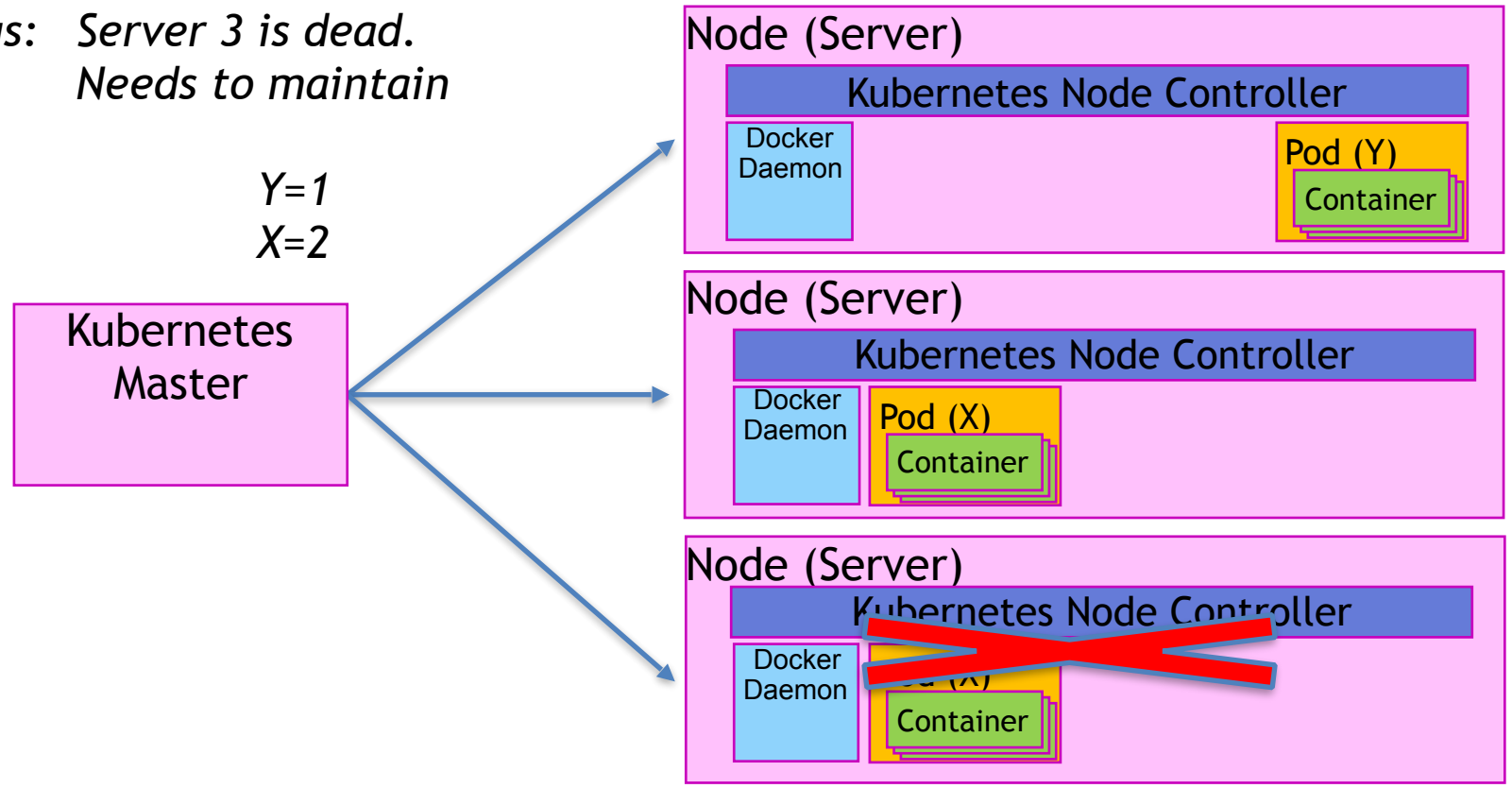
Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved
SAMSUNG SDS SAMSUNG

Example

Status: *Server 3 is dead.*
Needs to maintain

SLA:

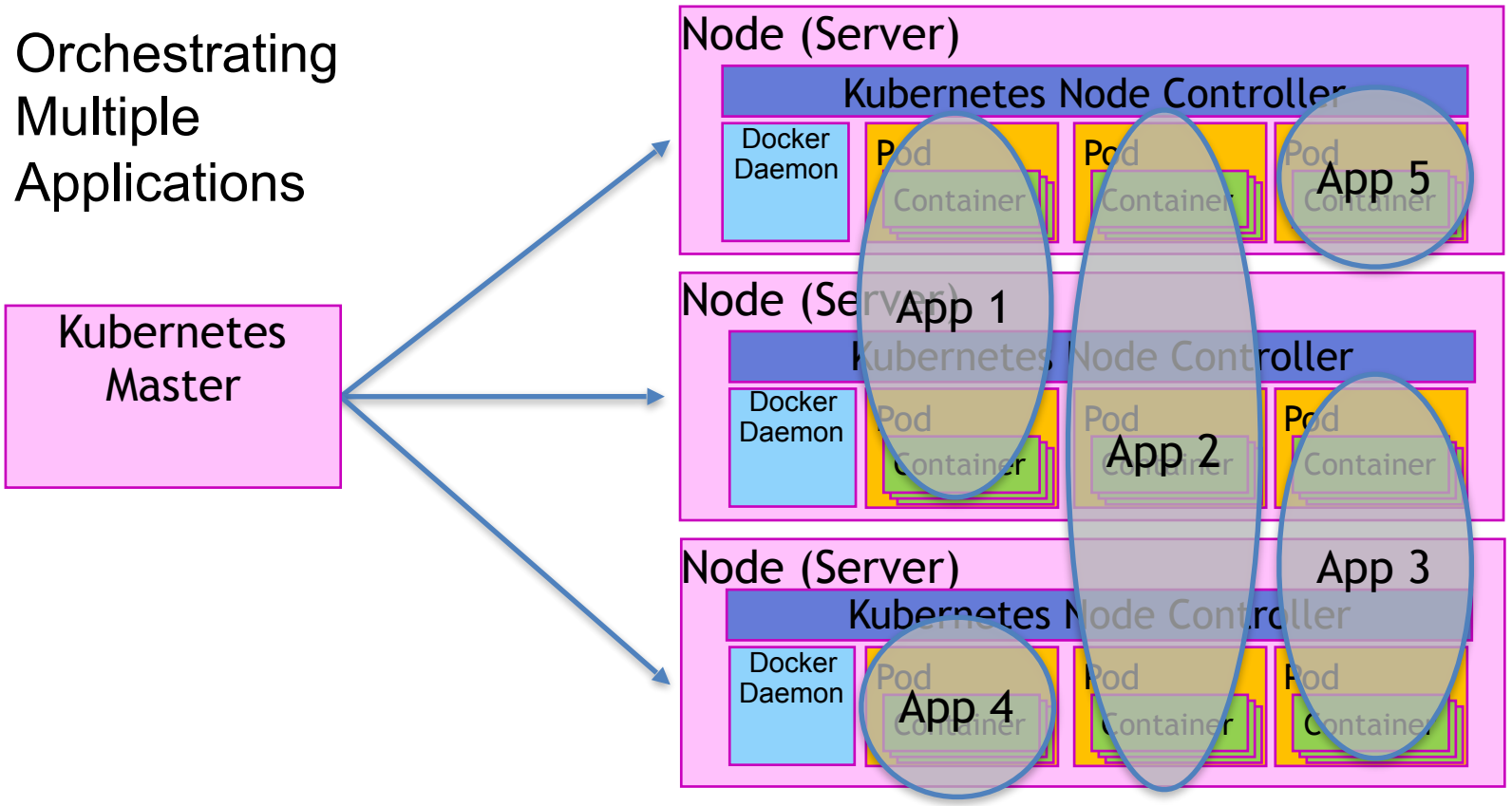
Y=1
X=2



Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Example

Orchestrating Multiple Applications



Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

~~Cybernetic Governor Helmsman 101~~

Goals and Ideals

Some Stories

Looking Ahead

Challenges

Why K8scale?

SAMSUNG SDS SAMSUNG

Kubernetes Special Interest Groups

- July 2015 – Kubernetes 1.0 Launch
- Post launch… interest in evolving the community by breaking into Special Interest Groups…
  - API Machinery
  - Autoscaling
  - Big Data
  - Cluster Ops
  - Network
  - Node
  - **Scalability**
  - Storage
  - Testing
  - UI
  - etc
- Aug 5, 2015 – first K8Scale meeting

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Kubernetes Special Interest Groups

- https://github.com/kubernetes/kubernetes/wiki/Special-Interest-Groups-(SIGs)

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

K8Scale Info

- Leads: Bob Wise (@countspongebob) and Joe Beda (@jbeda)
- Slack Channel: #sig-scale on slack.kubernetes.io
- Mailing List: kubernetes-sig-scale
- Meetings: Thursdays at 9am pacific. Contact Joe or Bob for invite.

- https://github.com/kubernetes/kubernetes/wiki/SIG-Scalability

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved
SAMSUNG SDS SAMSUNG

K8Scale Official Goals

- Scalability SLO’s
  - "API-responsiveness": 99% of all API calls return in less than 1s
  - "Pod startup time: 99% of pods (with pre-pulled images) start within 5s
- SLO’s should be valid for clusters that have
  - up to 1K nodes
  - up to 30K total pods (eg: 1K nodes, 30 pods-per-node)

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

K8Scale Ideals

- Larger scale
  - > 1000 nodes
  - > 100 pods per node
  - > 100 pods/sec scheduled

- Better practices
  - Ensure API server is responsive at scale
  - Ensure API server is available at scale
  - Avoid thundering herds
  - Don’t sacrifice security in the name of performance

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

K8Scale Ideals

- Broader support
  - Alternative providers
    - eg: AWS, Bare Metal, GCE
  - Alternative host OS's
    - eg: CoreOS, Debian, RHEL
  - Alternative container engines
    - eg: Docker, Hyper, RKT
  - Alternative networking
    - eg: Calico, Flannel, Weave

- Transparency
  - Make publicly accessible data-driven decisions
  - Allow others to reproduce your results
  - Use and improve community-owned tests / tools

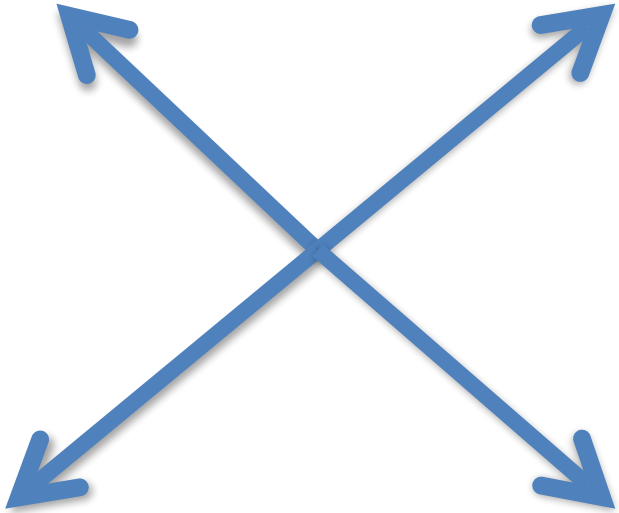
Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

Context – Scaling Dimensions

Node Count

Pod Rate



Pods/Node
(Node Size)

Latency

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

~~Cybernetic Governor Helmsman 101~~

~~Goals and Ideals~~

Some Stories

Looking Ahead

Challenges

Why K8scale?

SAMSUNG SDS SAMSUNG

Let’s Talk About The Density Test

- test/e2e/density.go
- schedule N pods-per-node
  - yields pod thoughput
- then schedule a few additional pods
  - yields pod latency when the cluster is under a given workload
- used to verify SLO’s

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

Tales from the trenches…

- We have to be data driven
- Intuitions can easily be wrong
- It's a bit too hard to replicate test setups from one team to another
- Here's an example…

https://github.com/kubernetes/kubernetes/issues/14216

"Stair-stepping in pods going from Pending to Running"

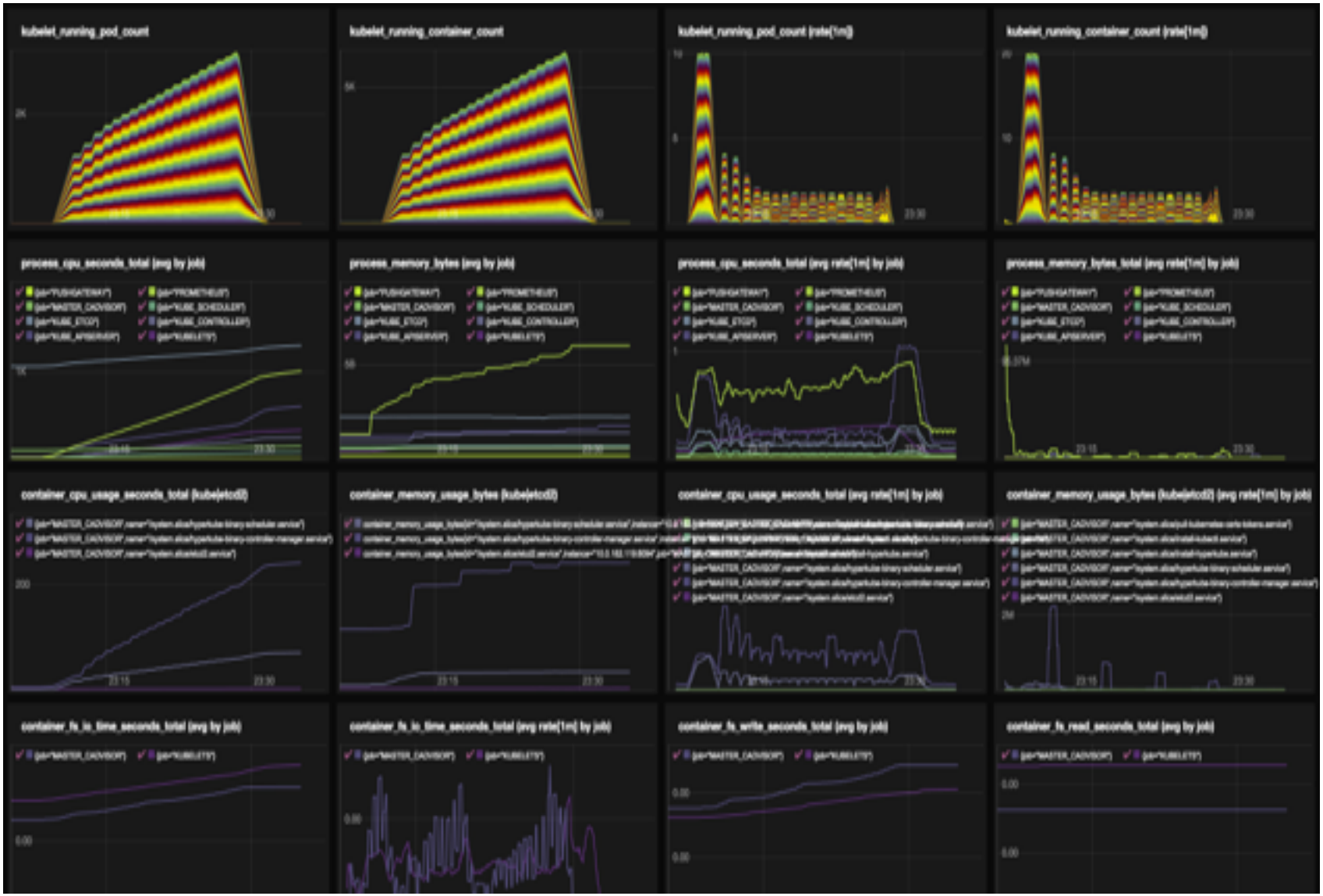
Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

Cluster Performance, Expected Result

SAMSUNG SDS SAMSUNG

Cluster Performance, Stairstepping



SAMSUNG SDS SAMSUNG

Stairstepping… Scale Zoom

The guessing…

- Scheduler has a bug
- Etcd is misbehaving
- Go garbage collection is firing
- QPS rate limits are causing backpressure
- Some performance difference between AWS and GCE instances is triggering bug (getting desperate now ☺️)
  - Only showing on AWS

We banged our heads on this one for weeks…

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

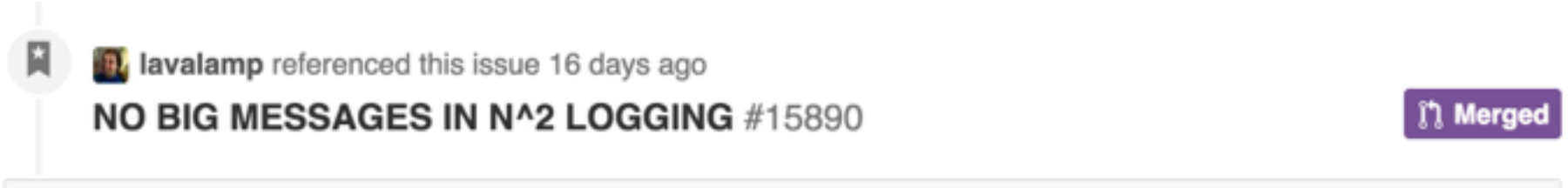
Cause…

- Scheduler logging was verbose
- Scheduler hits a buffer dump sync pause
- Nothing gets scheduled until the log buffer gets dumped
- Samsung/AWS setup was just enough bigger to hit the threshold

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Cause…

- Scheduler logging was verbose
- Scheduler hits a buffer dump sync point
- Nothing gets scheduled until the log buffer gets dumped
- Samsung/AWS setup was just enough bigger to hit the threshold



Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

BUT ACTUALLY

vcaputo commented on Nov 30, 2015

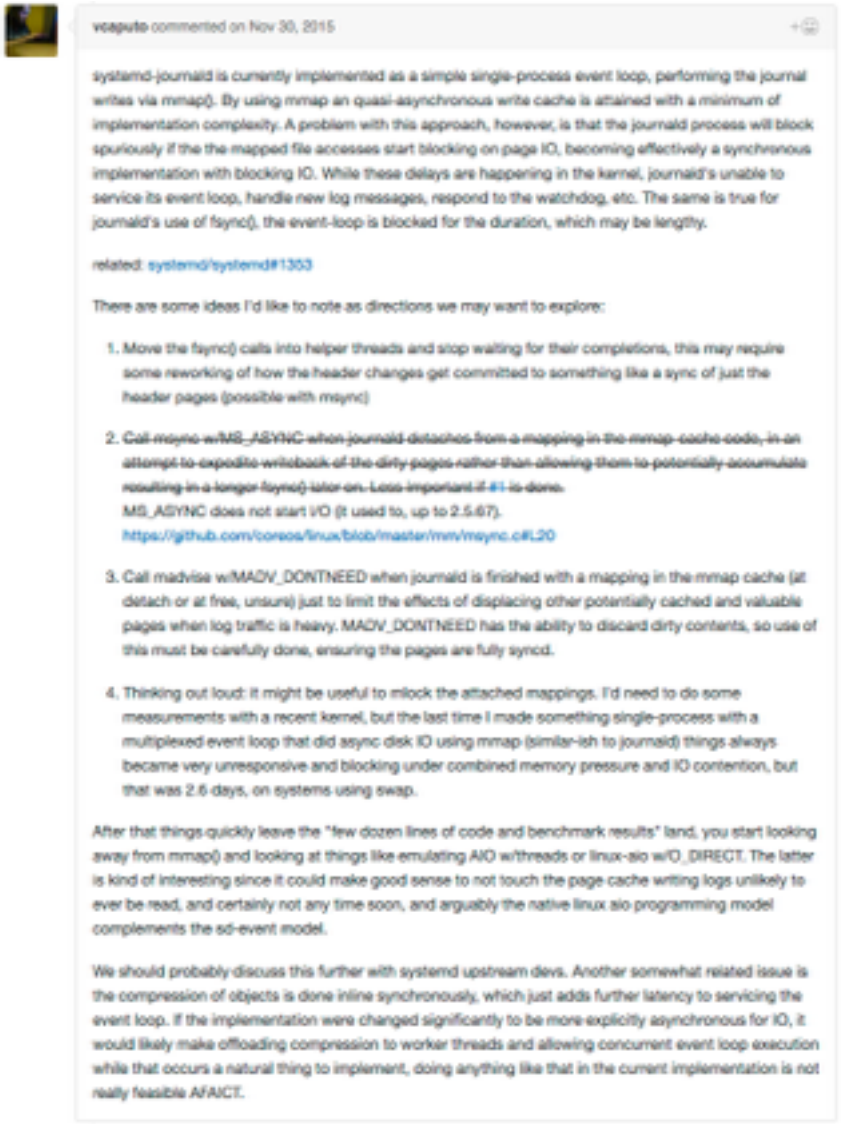
There are some issues with the current journald implementation, bypassing journald by either pointing the output to /dev/null or to (remote) syslog are the best options for log configuration in the unit file of chatty services.

countspongebob commented on Nov 30, 2015

@vcaputo "There are some issues with the current journald implementation"... could you elaborate?

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

BUT ACTUALLY



vcaputo commented on Nov 30, 2015

systemd-journald is currently implemented as a simple single-process event loop, performing the journal writes via mmap(). By using mmap an quasi-asynchronous write cache is attained with a minimum of implementation complexity. A problem with this approach, however, is that the journald process will block spuriously if the the mapped file accesses start blocking on page IO, becoming effectively a synchronous implementation with blocking IO. While these delays are happening in the kernel, journald's unable to service its event loop, handle new log messages, respond to the watchdog, etc. The same is true for journald's use of fsync(), the event-loop is blocked for the duration, which may be lengthy.

related: systemd/systemd#1353

There are some ideas I'd like to note as directions we may want to explore:

1. Move the fsync() calls into helper threads and stop waiting for their completions, this may require some reworking of how the header changes get committed to something like a sync of just the header pages (possible with msync)

2. ~~Call msync w/MS_ASYNC when journald detaches from a mapping in the mmap cache code, in an attempt to expedite writeback of the dirty pages rather than allowing them to potentially accumulate resulting in a longer fsync() later on. Less important if #1 is done.~~
   MS_ASYNC does not start I/O (it used to, up to 2.5.67).
   https://github.com/coreos/linux/blob/master/mm/msync.c#L20

3. Call madvise w/MADV_DONTNEED when journald is finished with a mapping in the mmap cache (at detach or at free, unsure) just to limit the effects of displacing other potentially cached and valuable pages when log traffic is heavy. MADV_DONTNEED has the ability to discard dirty contents, so use of this must be carefully done, ensuring the pages are fully synced.

4. Thinking out loud: it might be useful to mlock the attached mappings. I'd need to do some measurements with a recent kernel, but the last time I made something single-process with a multiplexed event loop that did async disk IO using mmap (similar-ish to journald) things always became very unresponsive and blocking under combined memory pressure and IO contention, but that was 2.6 days, on systems using swap.

After that things quickly leave the "few dozen lines of code and benchmark results" land, you start looking away from mmap() and looking at things like emulating AIO w/threads or linux-aio w/O_DIRECT. The latter is kind of interesting since it could make good sense to not touch the page cache writing logs unlikely to ever be read, and certainly not any time soon, and arguably the native linux aio programming model complements the sd-event model.

We should probably discuss this further with systemd upstream devs. Another somewhat related issue is the compression of objects is done inline synchronously, which just adds further latency to servicing the event loop. If the implementation were changed significantly to be more explicitly asynchronous for IO, it would likely make offloading compression to worker threads and allowing concurrent event loop execution while that occurs a natural thing to implement, doing anything like that in the current implementation is not really feasible AFAICT.

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

BUT ACTUALLY

- It was (also) systemd-journald
- This has been addressed in CoreOS alpha 1010.1.0
- https://github.com/coreos/bugs/issues/990
- https://github.com/coreos/bugs/issues/1162

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Notable Accomplishments

- Implement watch read-cache in API server
  - watch: a connection that will publish updates about the requested resources
  - a core part of kubernetes loosely-coupled design (complemented by polling)
  - kubelet instances “watch” for pods that match their node
  - kube-controller-manager will “watch” for replication controllers

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

Notable Accomplishments

- Implement watch read-cache in API server
  - previously: watch is a native etcd construct, proxy each request for a watch through to etcd
  - now: watch each resource type from etcd, store in a read-cache in API server, filter/select results and fan-out to individual watches

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

Notable Accomplishments

- Scheduling improvements
  - https://coreos.com/blog/improving-kubernetes-scheduler-performance.html
  - use more efficient math in scaling routine (https://github.com/kubernetes/kubernetes/pull/18170)
  - compute predicates in parallel (https://github.com/kubernetes/kubernetes/pull/18413)

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Notable Accomplishments

- kubelet efficiency gains
  - pod lifecycle event generator (https://github.com/kubernetes/kubernetes/commits/master/docs/proposals/pod-lifecycle-event-generator.md)
  - fix cadvisor leaking goroutines (https://github.com/kubernetes/kubernetes/issues/19633)

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Notable Accomplishments

- http://blog.kubernetes.io/2016/03/1000-nodes-and-beyond-updates-to-Kubernetes-performance-and-scalability-in-12.html

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

~~Cybernetic Governor Helmsman 101~~

~~Goals and Ideals~~

~~Some Stories~~

Looking Ahead

Challenges

Why K8scale?

SAMSUNG SDS SAMSUNG

K8scale 1.3 draft priorities

- 200k pods total
  - 2000 nodes at 100 pods/node
  - 400 nodes at 500 pods/node

- Reduce control plane network overhead
  - N connections per node -> use HTTP2 for re-use
  - TLS is expensive -> use go 1.6 to decrease
  - JSON encode/decode dominates API server load -> use protobuf (internally, still allow for JSON externally)
  - Client-side QPS limits over/under-optimize -> use server-side back pressure

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

K8scale 1.3 draft priorities

- Scheduler Optimization
  - parallelize as much as possible
  - use optimistic binding

- Etcd 3
  - variety of performance improvements anticipated
  - eg: gRPC instead of HTTP+JSON for watches
  - implementing against current interface
  - extending to take advantage of new etcd3 capabilities next

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

K8scale 1.3 draft priorities

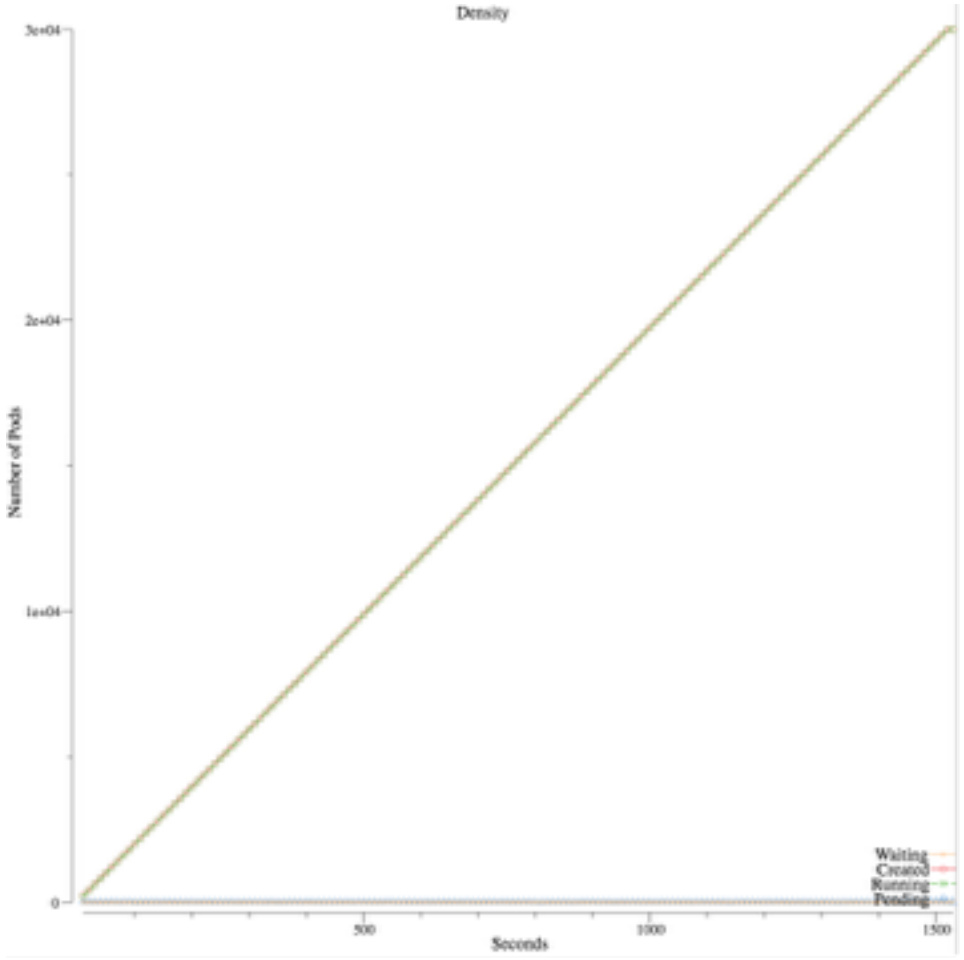
- Avoid thundering herds
  - a lot of components poll / re-list, revisit whether they need to poll
  - if they do, they need to jitter

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

eg: QPS limits over/under-optimize

- Scheduling is “fast enough” to keep up with new pods as they’re created at default QPS

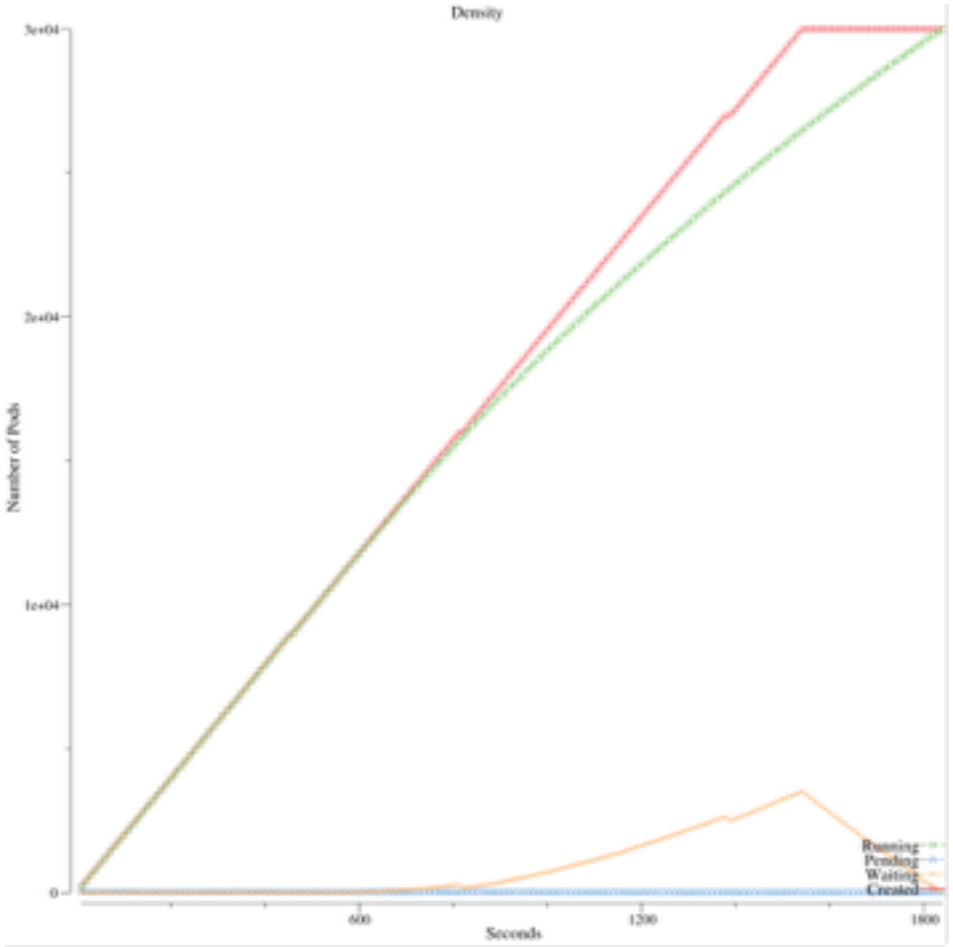


Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

eg: QPS limits over/under-optimize

- But raise the pod creation rate by tweaking kube-controller-manager's QPS setting and…



Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

eg: QPS limits over/under-optimize

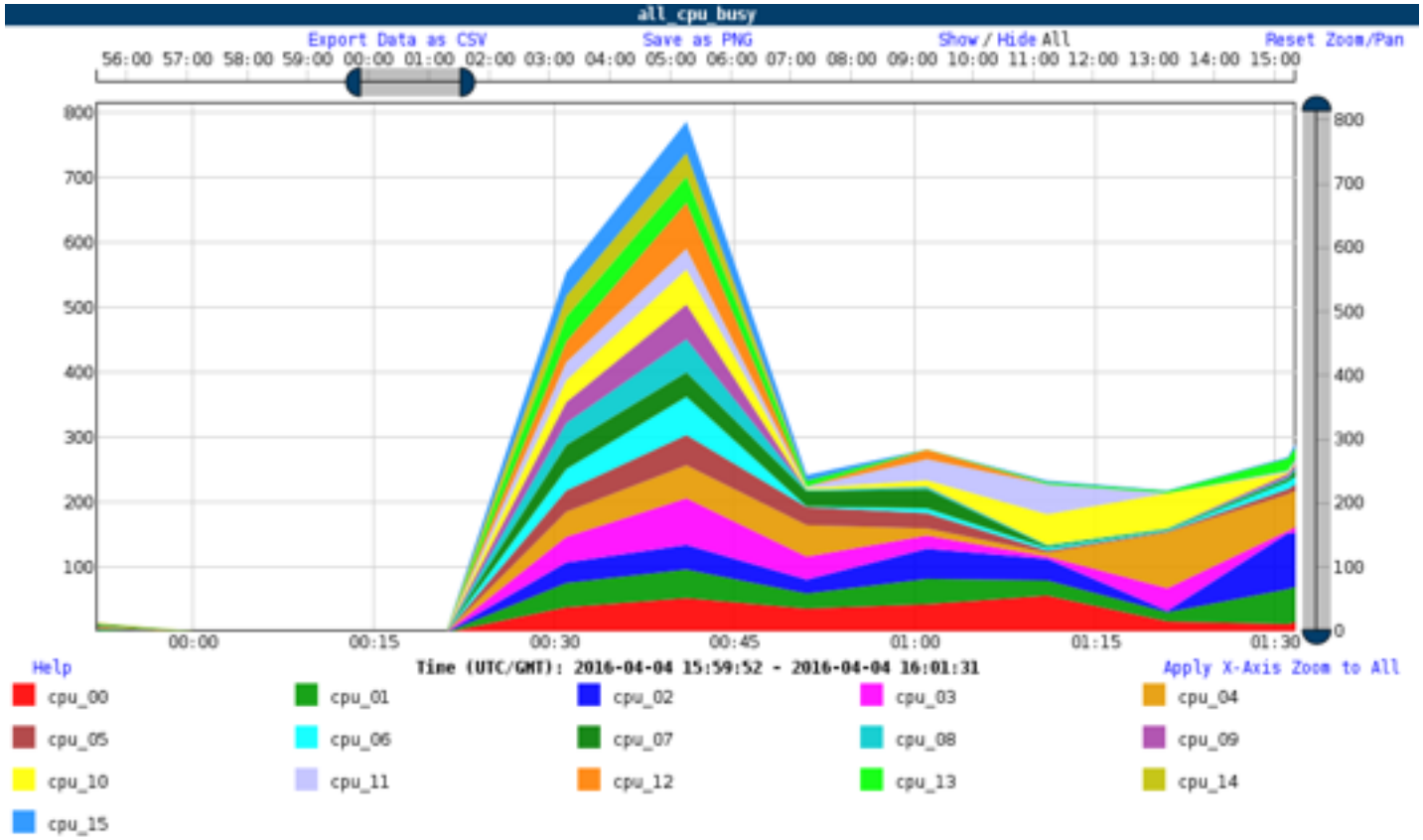
- https://github.com/kubernetes/kubernetes/issues/24408

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

eg: Avoid thundering herds

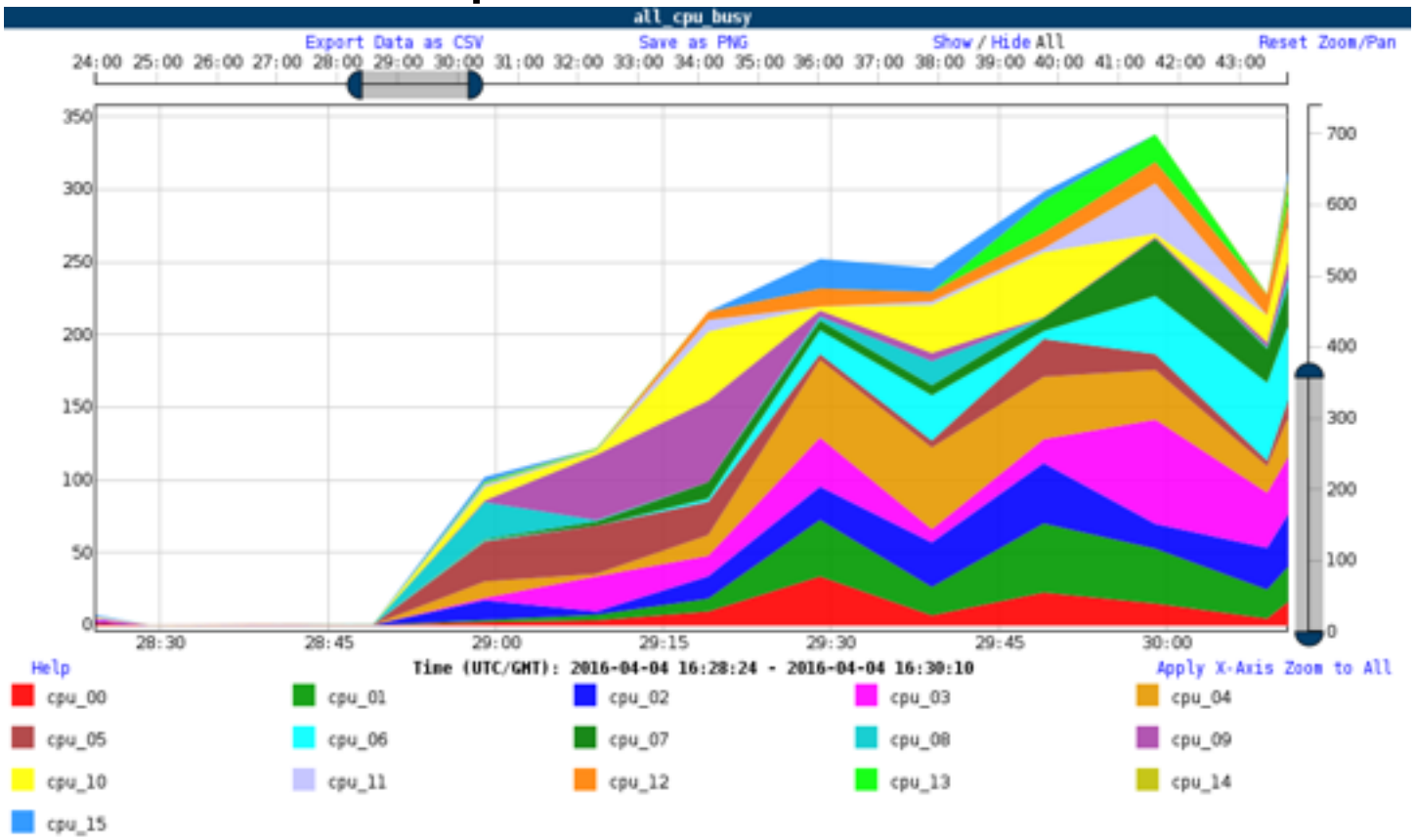
- kube-controller-manager starts all controllers at the same time



Copyright © 201 Samsung SDS Co., Ltd. All rights reserved

eg: Avoid thundering herds

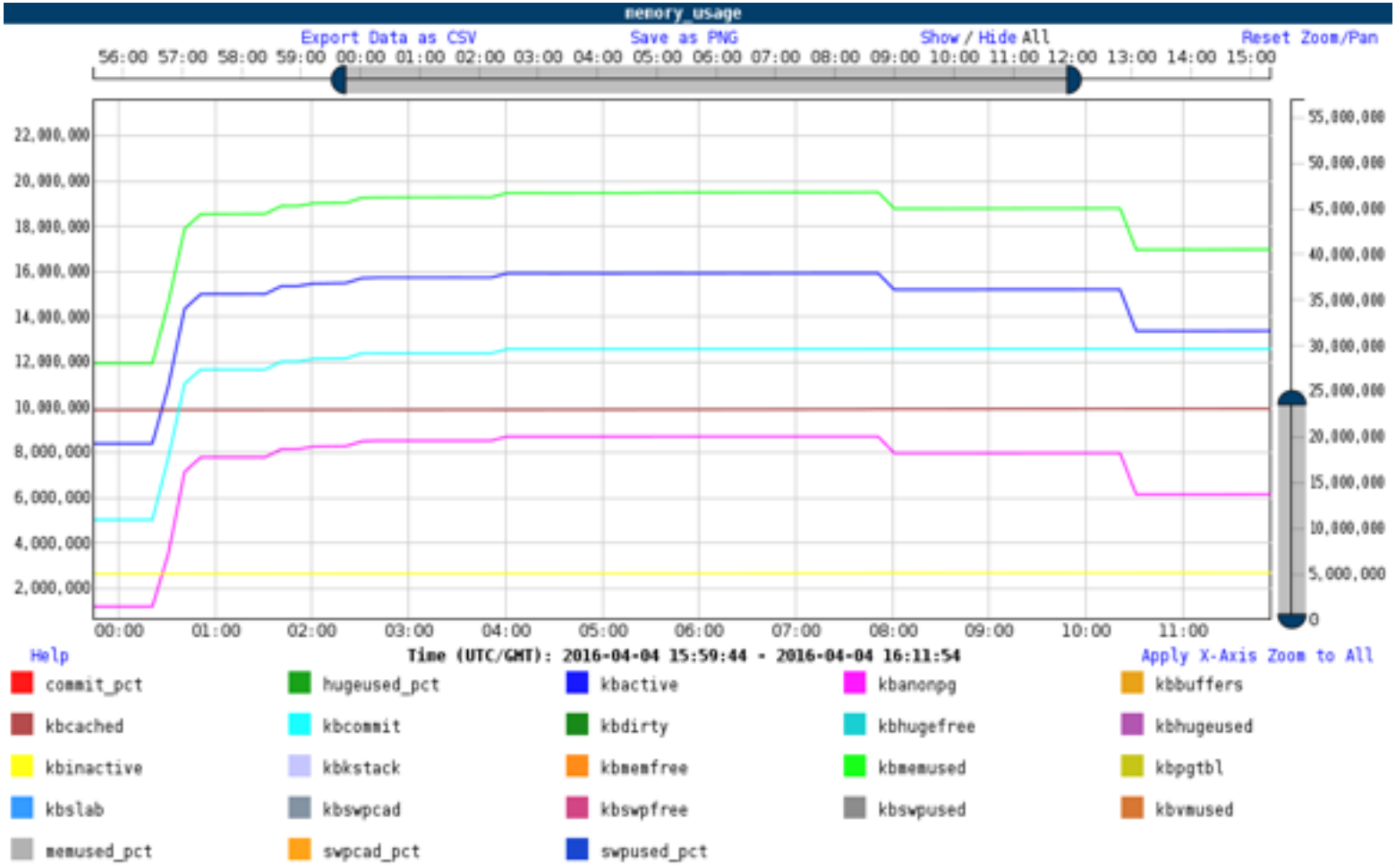
- kube-controller-manager introduces jitter to controller startup



Copyright © 201 Samsung SDS Co., Ltd. All rights reserved

eg: Avoid thundering herds

- kube-controller-manager starts all controllers at the same time

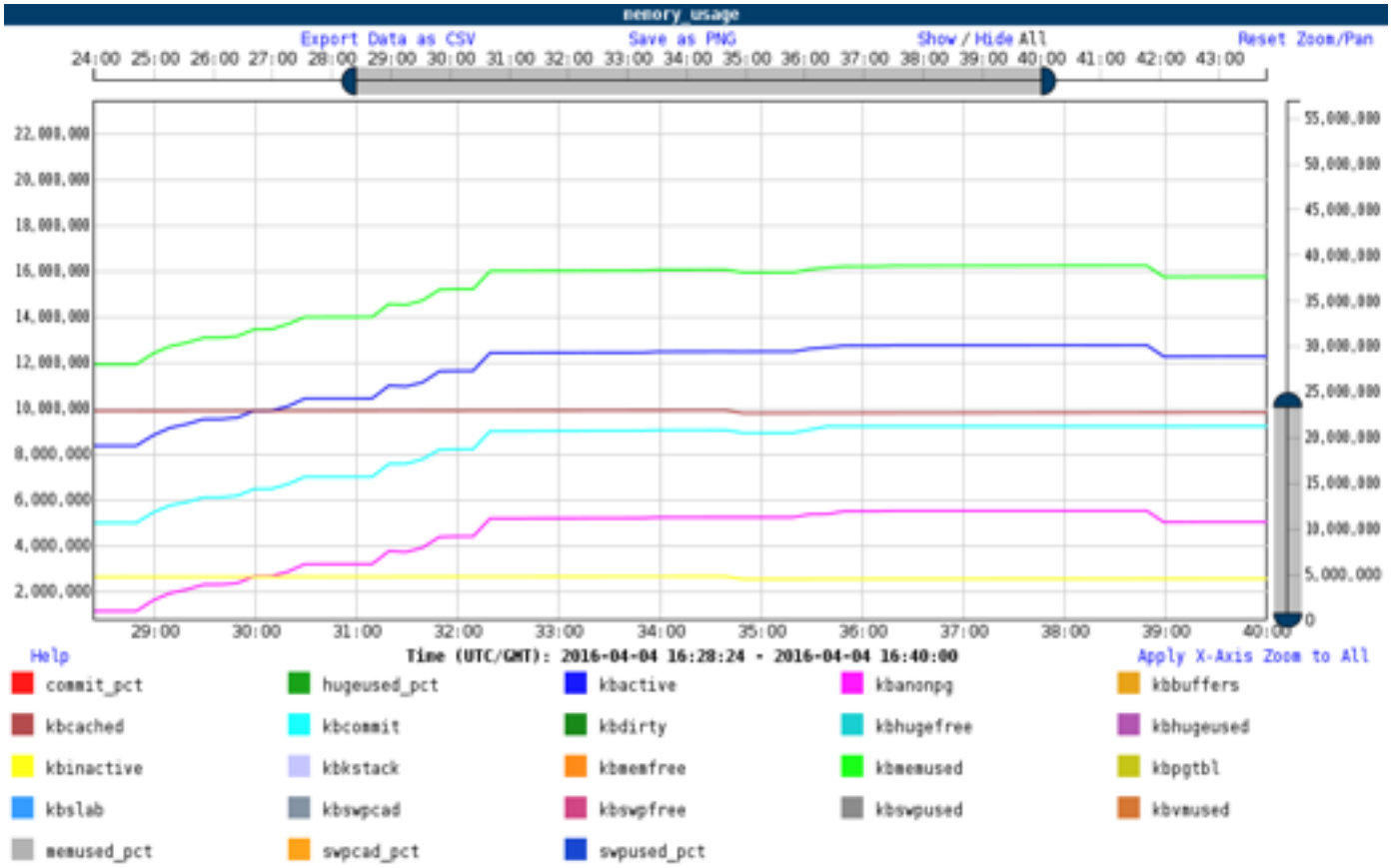


Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

eg: Avoid thundering herds

- kube-controller-manager introduces jitter to controller startup



Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

eg: Avoid thundering herds

- https://github.com/kubernetes/kubernetes/pull/23819

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

~~Cybernetic Governor Helmsman 101~~

~~Goals and Ideals~~

~~Some Stories~~

~~Looking Ahead~~

Challenges

Why K8scale?

SAMSUNG SDS SAMSUNG

Challenges - Deployment

- The out-of-the-box deploy experience leaves a lot to be desired
  - Not continuously tested on anything other than GCE and GKE
  - Difficult to come up with a single “cluster definition” that spells out cluster topology, infrastructure configuration, and component configuration
  - There are way too many tuning knobs that can have potentially unpredictable results
  - It is literally a pile of shell scripts

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Challenges - Deployment

- Our attempt at pushing deployment forward
  - https://github.com/samsung-cnct/kraken
  - AWS, vagrant
  - single terraform file for cluster topology
  - ansible role per component
  - use auto scaling groups to support 1K nodes
  - deploy across multiple AZ's
  - nodes provision themselves at boot (cloud-init lays down systemd units to run ansible-in-docker, which pulls from a git repo)

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

Challenges - Data Sharing

- Sharing performance data is highly desired by all, but as yet we don't have an agreed upon format to share time series data.
  - We really like prometheus
  - Some tests (eg: density) use the prometheus push gateway
  - There are dashboards that scrape prometheus metrics out of test logs (eg: http://perf-dash.k8s.io/)
  - This is a start, but we could really use something more fleshed out

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved
SAMSUNG SDS SAMSUNG

Challenges - Data Sharing

- Google's CI results are now publicly viewable
  - http://storage.googleapis.com/kubernetes-test-history/static/index.html
  - http://submit-queue.k8s.io/#/e2e

- Testing SIG is working on a federated testing proposal
  - community members submit results to GCS buckets
  - bots / dashboards updated to read from GCS

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

~~Cybernetic Governor Helmsman 101~~

~~Goals and Ideals~~

~~Some Stories~~

~~Looking Ahead~~

~~Challenges~~

Why K8scale?

SAMSUNG SDS SAMSUNG

Why K8scale?

- Kubernetes is 8 letters long

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

Why is Samsung involved in K8scale?

- We're actively helping internal and external customers on the path to cloud native computing
- We want "Google infrastructure for everyone else" (us!)
- We want very large clusters with cross application resource sharing
- We believe we can make a positive contribution to make this happen faster and better.
- We believe we need deep technical involvement to build/deploy/operate at scale
- We've been pushing the envelope on AWS

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved

SAMSUNG SDS SAMSUNG

Contact Info

- bob.wise@samsung.com
- bobwise on kubernetes.slack.com
- https://github.com/Samsung-CNCT

- aaron.c1@samsung.com
- spiffxp on kubernetes.slack.com
- https://github.com/spiffxp

(for K8scale also joe@0xBEDA.com)

Copyright © 2016 Samsung SDS Co., Ltd. All rights reserved